**UNIVERSIDADE DE SÃO PAULO**

**BACHARELADO EM CIÊNCIAS DE COMPUTAÇÃO**

**CLASSIFICAÇÃO DE IMAGENS: DIAGNÓSTICO DE NÍVEL DE ALZHEIMER**

**PAULO RICARDO JORDÃO MIRANDA, 10133456**

**PEDRO HENRIQUE MAGALHÃES CISDELI, 10289804**

**SÃO CARLOS**

**2022**

**Sumário**

[**Introdução**](#_c7yvx7icjwus) **3**

[Modelos utilizados](#_p015c7p6sgcj) 3

[Artigo Científico (Parte 3)](#_clksm99ha5gz) 3

[Métricas de avaliação](#_1ta2x0g1t7cm) 4

[**Desenvolvimento**](#_ulyz7jgvr0or) **4**

[Analisando o conjunto de dados](#_qow8mz2uinj2) 4

[Modelagem](#_xqr6tm9hwcfy) 5

[Pré-processamento](#_df4crup14v68) 6

[Treinamento](#_l00elkvf2u2n) 7

[**Conclusão**](#_de8040gciljv) **13**

[Considerando os modelos da Parte 2 (KNN, DT, SVM, MLP)](#_k3qazfd22pps) 15

[Análise geral](#_zhxf3prj36z2) 15

[Imagens de treinamento entre SVM e MLP](#_rerdi4tr5u2o) 15

[Melhor desempenho](#_e8dm9zonewwn) 15

[Considerando todos os modelos estudados](#_teoym99up7h6) 16

[Análise geral](#_wsqromxolhm6) 16

[Melhor desempenho](#_dofgrq8bm1ny) 16

# **Introdução**

O presente projeto tem como objetivo classificar imagens de ressonância magnética digitais de acordo com rótulos pré-estabelecidos, ou seja, trata-se de uma classificação (aprendizado supervisionado).

A base utilizada para o projeto consiste de um total de 6.400 imagens de tamanhos 128x128, em escala de cinza, e podem ser obtidas através do link: <https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset>. As imagens estão separadas por pastas cujo o nome é o rótulo das imagens ali contidas. Cada rótulo possui o seguinte nome com a respectiva quantidade de imagens:

* Classe - 1: *Mild Demented*, com 896 imagens
* Classe - 2: *Moderate Demented*, com 64 imagens
* Classe - 3: *Non Demented*, com 3200 imagens
* Classe - 4: *Very Mild Demented*, com 2240 imagens

## **Modelos utilizados**

Para a tarefa de classificação, serão utilizados os modelos KNN, Árvore de Decisão, SVM, Redes Neurais e Redes Neurais Convolucionais.

Como o conjunto de dados não é estruturado (imagens), o problema requer uma forma de extrair descritores das imagens para serem utilizadas na Árvore de decisão e no KNN. Já as Redes Neurais e Redes Neurais Convolucionais, são mais "flexíveis" e, portanto, não necessitam que sejam feitas esta extração.

## **Artigo Científico (Parte 3)**

Para a implementação do método aplicado na entrega 3 do projeto, foi utilizado como base o artigo científico [*A CNN Model: Earlier Diagnosis and Classification of Alzheimer Disease using MRI*](https://ieeexplore.ieee.org/document/9215402). Esse documento analisa a utilização de um modelo de Deep Learning em imagens provenientes de ressonância magnética para o auxílio na identificação da doença de Alzheimer. O modelo utilizado no estudo é uma rede neural de convolução e obteve uma acurácia de 99%.

## **Métricas de avaliação**

O conjunto de dados conta com um desbalanceamento, contando com muitas imagens da classe *Non Demented* do que imagens com classe *Moderate Demented*, por exemplo. Por este motivo, é necessário utilizar a métrica correta para contornar o problema do desbalanceamento, uma vez que, não podemos utilizar técnicas de amostragem com este conjunto de dados. Assim, uma das métricas que serão utilizadas aqui são Acurácia e AUC (*Area Under Curve*).

# **Desenvolvimento**

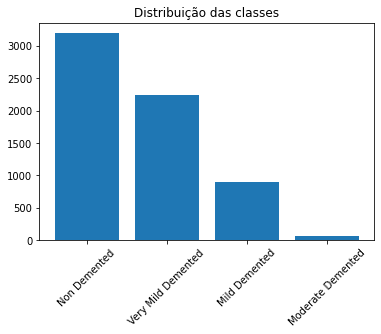
O desenvolvimento do projeto foi realizado na plataforma do Google Colab com o objetivo de facilitar o desenvolvimento em conjunto. Assim, foi necessário realizar o procedimento para poder acessar os diretórios do Google Drive onde se encontrava a base de dados. Ainda, foram utilizadas as bibliotecas *SKLearn*, para utilização dos modelos de Rede Neural, KNN, Árvore de Decisão e SVM como também o uso de funções para métricas de avaliação (acurácia e AUC); *SKImage*, para auxiliar no carregamento das imagens, extrair características das imagens e função para adicionar ruídos às imagens; *MatplotLib* para plotar gráficos; *Numpy* para a manipulação de estruturas como matrizes e vetores; e o *Pandas* para facilitar a visualização de métricas, criando tabelas com os dados das mesmas.

## 

## **Analisando o conjunto de dados**

Ao carregar as imagens que serão utilizadas no projeto, é uma boa prática verificar a distribuição dos dados. Como, neste caso, o projeto tem como objetivo classificar imagens, deve-se visualizar a distribuição das classes. A figura abaixo mostra a distribuição das classes *Mild Demented*, *Moderated Demented*, *Non Demented* e *Very Mild Demented*.

Figura 1: Distribuição das classes



Perceba que há uma diferença significativa na quantidade de imagens de uma classe para a outra. Isso fica ainda mais evidente com a classe *Moderate Demented* na compõe uma parte muito pequena (1%) do conjunto de dados. Isso irá impactar na escolha de métricas para avaliação dos modelos, como também na reprodução desta característica do problema no momento do treinamento dos modelos, ou seja, no momento de treinar os modelos candidatos, é necessário refletir essa característica (manter a proporcionalidade das classes).

## **Modelagem**

Nesta seção serão apresentados o pré-processamento realizado no conjunto de dados, a divisão do conjunto de dados em conjuntos de teste e treinamento, os modelos treinados com suas respectivas variações de hiperparâmetros, e o método utilizado para escolher o melhor de cada modelo.

### 

### **Pré-processamento**

Os modelos como KNN, Árvore de Decisão e SVM, não conseguem ter como sua entrada uma imagem (uma matriz), logo, é necessário executar algum procedimento para transformar estes dados não estruturados (imagens) em dados estruturados (tabela). Para tal, foi utilizado o *Local Binary Pattern*, tanto nas imagens de treinamento quanto nas imagens de teste.

Figura 2: Local Binary Pattern aplicado a uma imagem do conjunto de dados

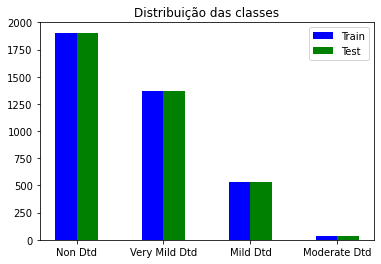
Em um cenário ideal, as imagens utilizadas no treinamento e no teste de um modelo tem especificações (altura, largura, resolução) bem definidas, porém, no mundo real o cenário é diferente: as imagens contém resoluções diferentes das que foram utilizadas para o treinamento e no teste, podem ter altura e largura também diferentes. Para contornar este problema, foram geradas imagens com diferentes níveis de ruído nos conjuntos de treinamento e teste. O objetivo é tentar fazer com que o modelo consiga generalizar bem e conseguir classificar, de maneira correta, uma imagem mesmo em condições não ideais.

Deste modo, foram criados mais 4 pares (conjunto de dados de treinamento e teste) de conjuntos de dados:

* Imagens com Local Binary Pattern com ruído aleatório seguindo uma Distribuição Gaussiana com variância de 0,01
* Imagens com Local Binary Pattern com com ruído aleatório seguindo uma Distribuição Gaussiana com variância de 0,001
* Imagens com ruído aleatório seguindo uma Distribuição Gaussiana com variância de 0,01
* Imagens com com ruído aleatório seguindo uma Distribuição Gaussiana com variância de 0,001

Quanto maior for a variância, neste caso, mais ruído a imagem vai ter.

Por fim, a distribuição das classes dos conjuntos de treinamento e teste é mostrado na figura abaixo.

Figura 3: distribuição das classes no conjunto de treino (*Train*) e teste (*Test*)

### **Treinamento**

Os modelos escolhidos para a classificação das imagens foram o KNN, SVM, Árvores de Decisão, Redes Neurais, mais especificamente a Rede Neural Multicamadas Perceptron e Redes Neurais Convolucionais.

Todos os modelos acima citados têm hiperparâmetros e para escolher os melhores hiperparâmetros dentre um conjunto pré-definido, foi utilizado a abordagem do *Grid Search* com *Stratified K-Fold* *Cross Validation*. Ainda, cada modelo foi treinado nas três variações do conjunto de dados: sem ruído e com ruído.

* **KNN**

Os hiperparâmetros que foram utilizados na busca da melhor combinação de hiperparâmetros foram:

knn\_hiperparams = {

'n\_neighbors': [1, 3, 5, 7, 13, 23],

'p': [1, 2, 3]

}

Os resultados obtidos (conjuntos de teste) do KNN treinado com as imagens sem ruído são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.530078 | 0.339453 | 0.341016 |
| AUC | 0.539670 | 0.500000 | 0.498486 |

Os resultados obtidos (conjuntos de teste) do KNN treinado com as imagens com ruído 0,01 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.339453 | 0.519922 | 0.432031 |
| AUC | 0.500000 | 0.539613 | 0.499202 |

Os resultados obtidos (conjuntos de teste) do KNN treinado com as imagens com ruído 0,001 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.350391 | 0.339453 | 0.510547 |
| AUC | 0.494563 | 0.500000 | 0.527774 |

* **Árvore de decisão**

Os hiperparâmetros que foram utilizados na busca da melhor combinação de hiperparâmetros foram:

decision\_tree\_hiperparams = {

'splitter': ['best', 'random'],

'max\_depth': [2, 4, 5, 8, 10],

'max\_leaf\_nodes': [2, 4, 6, 8, 10, 20]

}

Os resultados obtidos (conjuntos de teste) da árvore de decisão treinado com as imagens sem ruído são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.512109 | 0.339453 | 0.363281 |
| AUC | 0.538356 | 0.500000 | 0.480461 |

Os resultados obtidos (conjuntos de teste) da árvore de decisão treinado com as imagens com ruído 0,01 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.339453 | 0.509375 | 0.339453 |
| AUC | 0.500000 | 0.526035 | 0.500000 |

Os resultados obtidos (conjuntos de teste) da árvore de decisão treinado com as imagens com ruído 0,001 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.339453 | 0.430859 | 0.507031 |
| AUC | 0.500000 | 0.503805 | 0.526628 |

* **HOG - SVM**

Os hiperparâmetros que foram utilizados na busca da melhor combinação de hiperparâmetros foram:

hogify = HogTransformer(

pixels\_per\_cell=(14, 14),

cells\_per\_block=(2,2),

orientations=9,

block\_norm='L2-Hys'

)

Os resultados obtidos (conjuntos de teste) do modelo HOG - SVM treinado com as imagens sem ruído são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.701562 | 0.559766 | 0.666797 |
| AUC | 0.692319 | 0.587426 | 0.663912 |

Os resultados obtidos (conjuntos de teste) do modelo HOG - SVMtreinado com as imagens com ruído 0,01 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.568359 | 0.581641 | 0.575391 |
| AUC | 0.613187 | 0.617188 | 0.626060 |

Os resultados obtidos (conjuntos de teste) do modelo HOG - SVM treinado com as imagens com ruído 0,001 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.667188 | 0.557031 | 0.671484 |
| AUC | 0.665979 | 0.594738 | 0.669177 |

* **Rede Neural**

Os hiperparâmetros que foram utilizados na busca da melhor combinação de hiperparâmetros foram:

mlp\_hiperparams = {

'hidden\_layer\_sizes': [(5, 3, 2, 4), (5, 5, 4), (10, 6, 4)],

'max\_iter': [100]

}

Os resultados obtidos (conjuntos de teste) do modelo de rede neural treinado com as imagens sem ruído são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.713281 | 0.605078 | 0.692969 |
| AUC | 0.712520 | 0.667433 | 0.707478 |

Os resultados obtidos (conjuntos de teste) do modelo de rede neuraltreinado com as imagens com ruído 0,01 são mostrados na tabela abaixo:

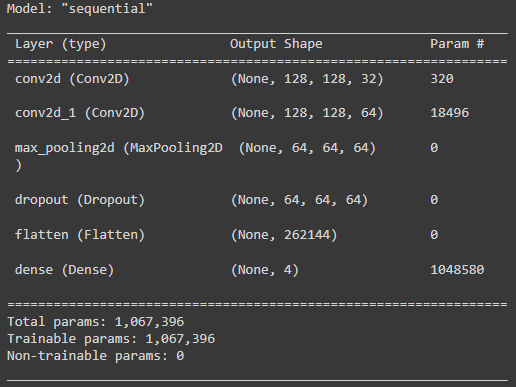
|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.713281 | 0.694141 | 0.715625 |
| AUC | 0.665995 | 0.675296 | 0.673142 |

Os resultados obtidos (conjuntos de teste) do modelo de rede neural treinado com as imagens com ruído 0,001 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.719141 | 0.559766 | 0.758984 |
| AUC | 0.664418 | 0.653986 | 0.708122 |

* **Rede Neural Convolucional**

A arquitetura da Rede Neural Convolucional utilizada foi a mesma descrita no artigo:



Os resultados obtidos (conjuntos de teste) do modelo de rede neural convolucional treinado com as imagens sem ruído são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.963672 | 0.820703 | 0.950391 |
| AUC | 0.984866 | 0.907450 | 0.981045 |

Os resultados obtidos (conjuntos de teste) do modelo de rede neural convolucionaltreinado com as imagens com ruído 0,01 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.964063 | 0.926953 | 0.960547 |
| AUC | 0.932558 | 0.812616 | 0.923108 |

Os resultados obtidos (conjuntos de teste) do modelo de rede neural convolucional treinado com as imagens com ruído 0,001 são mostrados na tabela abaixo:

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.958594 | 0.911719 | 0.955859 |
| AUC | 0.988495 | 0.977624 | 0.987585 |

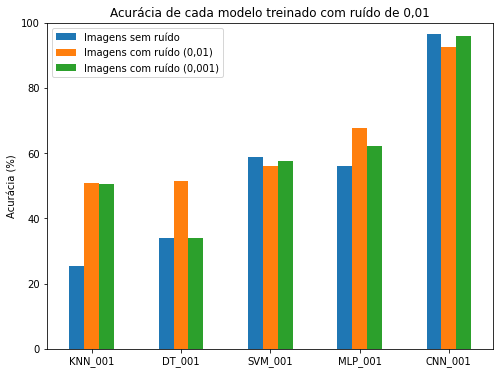
# 

# **Conclusão**

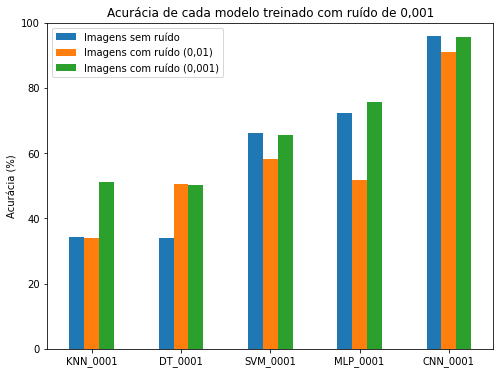
* Resultados obtidos no treinamento utilizando **imagens sem ruído**:

# 

* Resultados obtidos no treinamento utilizando **com ruído de 0,01**:



* Resultados obtidos no treinamento utilizando **com ruído de 0,001**:



## 

## **Considerando os modelos da Parte 2 (KNN, DT, SVM, MLP)**

### **Análise geral**

Verifica-se, portanto, que, por meio da análise dos gráficos e dos dados obtidos, os modelos de máquina de vetores de suporte (HOG-SVM) e redes neurais (MLP) obtiveram o melhor desempenho quando comparados com os outros estudados. Resultado já esperado devido às características da aplicação.

### **Imagens de treinamento entre SVM e MLP**

Ao comparar o melhor tipo de imagem de treinamento para os dois modelos de melhor desempenho, é possível notar que, ao realizar o processo de treino com ruído gaussiano de 0.01, a acurácia apresentada teve menor desigualdade para os três tipos de imagens de entrada.

### **Melhor desempenho**

Por meio da análise dos três métodos de treinamento das redes neurais, nota-se que a maior acurácia foi obtida por meio do treino com ruído de 0.001 e entrada sem nenhum ou 0.001, contudo, esse modelo apresenta resultados muito inferiores para o outro tipo de entrada. Dessa forma, é possível concluir que o método mais eficiente e constante para os três tipos de entrada foi a **rede neural treinada por imagens com ruído gaussiano de 0.01.**

|  | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| MLP | 0.713281 | 0.605078 | 0.692969 |
| **MLP\_001** | **0.713281** | **0.694141** | **0.715625** |
| MLP\_0001 | 0.719141 | 0.559766 | 0.758984 |

## **Considerando todos os modelos estudados**

### **Análise geral**

Verifica-se, portanto, que, por meio da análise dos gráficos e dos dados obtidos, o modelo de rede neural convolucional (CNN) obteve o melhor desempenho quando comparados com os outros estudados. Resultado já esperado devido às características da aplicação.

### **Melhor desempenho**

Por meio da análise dos três métodos de treinamento das redes neurais convolucionais, nota-se que a maior **acurácia média** foi obtida por meio do treino com ruído de 0.01, contudo, ao considerar o **AUC** e a **Acurácia**, é possível notar que o modelo treinado por imagens com ruído de 0.001 obteve resultados mais constantes e melhores. Dessa forma, é possível concluir que o método mais eficiente e constante para os três tipos de entrada foi a **rede neural convolucional treinada com imagens com ruído gaussiano de 0.001**.

| **CNN\_0001** | Imagens sem ruído | Imagens com ruído (0,01) | Imagens com ruído (0,001) |
| --- | --- | --- | --- |
| Acurácia | 0.958594 | 0.911719 | 0.955859 |
| AUC | 0.988495 | 0.977624 | 0.987585 |

**Bibliografia**

DUBEY, Sarvesh. **Alzheimer's Dataset**. 2020. Disponível em: <https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images>. Acesso em: 20 abr. 2022.

GAION, João Pedro de Barros Fernandes. **Doença de Alzheimer**: saiba mais sobre a principal causa de demência no mundo. saiba mais sobre a principal causa de demência no mundo. 2020. Disponível em: <https://www.informasus.ufscar.br/doenca-de-alzheimer-saiba-mais-sobre-a-principal-causa-de-demencia-no-mundo/>. Acesso em: 20 abr. 2022.

ALZHEIMER'S ASSOCIATION. **Alzheimer e demência no Brasil**. 20--?. Disponível em: <https://www.alz.org/br/demencia-alzheimer-brasil.asp>. Acesso em: 20 abr. 2022.

NAZARÉ, Thiago Santana de, et. al. **Deep Convolutional Neural Networks and Noisy Images**. Disponível em:<https://sites.icmc.usp.br/moacir/papers/Nazare_CIARP2017_DNN-Noise.pdf>. Acesso em: 22 abr. 2022.

PADOVESE, Bruno Tavares. **Suporte ao Diagnóstico da Doença de Alzheimer a partir de imagens de Ressonância Magnética**. Disponível em: <https://repositorio.unesp.br/bitstream/handle/11449/151042/padovese_bt_me_sjrp.pdf?sequence=3&isAllowed=y>. Acesso em: 25 maio 2022.

SALEHI, Ahmad Waleed; BAGLAT, Preety; SHARMA, Brij Bhushan; GUPTA, Gaurav; UPADHYA, Ankita. **A CNN Model**: **Earlier Diagnosis and Classification of Alzheimer Disease using MRI**. 2019. 6 f. Shoolini University, Solan, 2019. Disponível em: <https://ieeexplore.ieee.org/document/9215402>. Acesso em: 30 junho 2022.